doi:10.3969/j.issn.1001-4616.2022.01.017

改进粒子速度和位置更新公式的粒子群优化算法

李二超,高振磊

(兰州理工大学电气工程与信息工程学院,甘肃 兰州 730050)

[摘要] 针对粒子群优化算法求解精度低、局部搜索能力差、进化后期收敛速度慢等问题,本文提出一种改进粒子速度和位置更新公式的粒子群优化算法(particle swarm optimization algorithm with improved particle velocity and position update formula,IPSO-VP). IPSO-VP 算法提出一种自适应粒子速度和位置更新策略,采用基于 Logistic 混沌呈非线性变化的惯性权重,以此来加快算法的收敛速度、平衡算法的全局和局部搜索能力、提高收敛精度. 最后将本文所提算法与 6 个改进粒子群算法在 12 个测试函数上进行寻优比较,结果表明,本文所提算法在收敛速度和寻优精度方面均优于其他 6 种改进算法.

[关键词] 粒子群优化,自适应,Logistic 混沌,收敛速度,寻优精度
[中图分类号]TP273 [文献标志码]A [文章编号]1001-4616(2022)01-0118-09

Particle Swarm Optimization Algorithm With Improved Particle Velocity and Position Update Formula

Li Erchao,Gao Zhenlei

(College of Electrical and Information Engineering,Lanzhou University of Technology,Lanzhou 730050,China)

Abstract:Particle swarm optimization algorithm with improved particle velocity and position update formula(IPSO-VP) is proposed to solve the problems of low solution precision,poor local search ability and slow convergence rate in the later stage of evolution. IPSO-VP algorithm proposes an adaptive particle velocity and position update strategy,which adopts the inertia weight based on Logistic chaos,which is nonlinear,to accelerate the convergence rate,balance the global and local search ability of the algorithm,and improve the convergence precision. Finally,the proposed algorithm is compared with six improved particle swarm optimization algorithms on twelve test functions. Simulation results show that the proposed algorithm is superior to the other six improved particle swarm optimization algorithms in terms of convergence rate and optimization precision.

Key words:particle swarm optimization,adaptive,Logistic chaos,convergence rate,optimization precision

粒子群优化(Particle Swarm Optimization,PSO)算法是美国心理学家 Kennedy 和电气工程师 Eberhart 受鸟类和鱼类觅食行为启发而提出的一种基于群体智能理论的演化计算技术[1]. 该算法因其计算内存需求少、控制参数少等优点得到广泛关注,并被应用于求解诸多实际优化问题,如路径规划[2]、优化调度[3]、参数辨识[4]、图像分割[5]等.

然而,标准 PSO 算法存在一些固有缺陷,如搜索精度低、局部搜索能力差,尤其求解复杂非线性多峰函数时容易陷入局部极小解出现“早熟”收敛[6]. 针对以上缺陷,国内外学者做了大量改进研究,其中对粒子速度和位置更新公式的改进是研究热点之一. Shi 等[7]首次在速度更新项中引入随迭代次数线性递减的惯性权重 w,以更好平衡算法全局开发能力与局部勘探能力. Liu 等[8]指出 PSO 算法求解过程本身就是一个非线性过程,w 呈非线性变化能更好改善算法性能,因此他提出基于 Logistic 混沌映射的非线性变化的 w. 对 w 的改进还有基于余弦函数[9]和基于 Sigmoid 函数非线性变化的 w[10]等. Deep 等[11]将粒子速度更新公式中个体最优和群体最优替换为二者线性组合,以此来增大粒子的搜索空间,从而使算法有更多可

收稿日期:2021-08-31.
基金项目:国家自然科学基金项目(61763026).
通讯作者:李二超,博士,教授,博士生导师,研究方向:多目标优化、人工智能、进化计算. E-mail:lecstarr@163.com

能搜索到全局最优解. 胡旺等[12]提出一种简化 PSO 算法,尝试舍弃粒子速度项,由个体最优粒子和群体最优粒子来引导粒子更新,实验结果表明这种改进极大提高了算法性能. Liu 等[13]在简化 PSO 算法[12]基础上,引入粒子"维数信息"概念,将粒子位置更新分解为 3 种模式,3 种模式分别具有改善全局开发能力、改善局部勘探能力、提高算法收敛速度的功能,并基于概率来选择相应模式更新粒子位置. 陆松建等[14]在文献[11-12]的基础上提出一种均值简化粒子群算法,该算法舍弃速度更新项,将文献[11]对速度更新项的改进思想用于对位置更新的改进. Zhang 等[15]提出 2 种不同的位置更新公式,并在算法不同迭代时期选择相应公式,以此来改善算法性能. Liu 等[8]在文献[15]的研究基础上,在当前粒子适应度值与种群中所有粒子的平均适应度值之间建立一种数学关系 p_i,提出一种自适应位置更新策略来更好平衡算法的全局开发能力和局部勘探能力. Cheng 等[16]基于社会学习的思想,改进速度和位置更新机制,粒子速度和位置更新时不再向个体最优和群体最优学习,而向一些优于自身的个体学习.

本文结合以上分析,借鉴文献[8,11-13]的改进思想,提出一种新的粒子位置和速度自适应更新策略,引用文献[8]中的自适应判定机制,并将文献[13]提出的粒子维度信息引入粒子速度更新公式,将其与其他 6 种改进 PSO 算法在 12 个不同类型的测试函数上进行寻优测试,证明了本文所提算法的有效性.

1 相关工作

1.1 标准粒子群优化算法

标准 PSO 算法中,粒子通过学习自身历史经验(pbest)与群体经验(gbest)寻找最优粒子. 对于求解变量为 $X=\{x_1,x_2,\cdots,x_D\}$、目标函数为 $\min\{f(x)\}$ 的优化问题,标准 PSO 算法粒子更新公式为式(1)和式(2).

$$v_{id}(t+1)=wv_{id}(t)+c_1r_1(\text{pbest}_{id}-x_{id}(t))+c_2r_2(\text{gbest}_d-x_{id}(t)), \tag{1}$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1), \tag{2}$$

式中,$v_{id}(t+1)$和 $x_{id}(t+1)$分别为粒子 i 在 $t+1$ 代的速度和位置;w 是惯性权重,标准 PSO 算法中 w 随迭代次数线性递减;c_1 和 c_2 为学习因子,通常取值为 2;r_1 和 r_2 是[0,1]均匀分布的随机数.

1.2 粒子速度更新公式的改进

文献[11]提出一种均值粒子群优化(MeanPSO)算法,即利用个体最优和群体最优的线性组合 $\frac{\text{pbest}_{id}+\text{gbest}_d}{2}$ 和 $\frac{\text{pbest}_{id}-\text{gbest}_d}{2}$ 分别替换公式(1)中 pbest_{id} 和 gbest_d,得到新的粒子速度更新公式,为式(3).

$$v_{id}(t+1)=wv_{id}(t)+c_1r_1\left(\frac{\text{pbest}_{id}+\text{gbest}_d}{2}-x_{id}(t)\right)+c_2r_2\left(\frac{\text{pbest}_{id}-\text{gbest}_d}{2}-x_{id}(t)\right). \tag{3}$$

MeanPSO 算法和 PSO 算法中粒子运动过程如图 1 所示.

从图 1 可看出,MeanPSO 算法中粒子搜索区间更广,使得算法在进化前期有更大可能搜索到全局最优解[14].

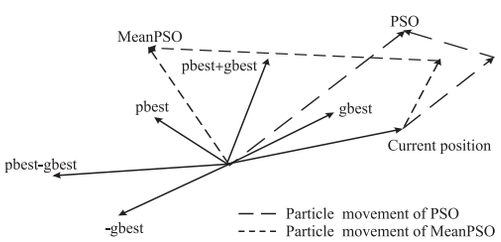

图 1 MeanPSO 算法和 PSO 算法粒子进化过程

Fig. 1 Particle evolution process of MeanPSO algorithm and PSO algorithm

1.3 粒子位置更新公式的改进

文献[13]提出一种带有平均维度信息的分层简化粒子群优化(PHSPSO)算法,PHSPSO 算法舍弃 PSO 算法中粒子速度更新项,并引入平均维度信息概念,即每个粒子所有维信息的平均值,计算公式为式(4). 同时 PHSPSO 算法将粒子位置更新公式分解为 3 种模式,分别为式(5)、(6)、(7).

$$p_{ad}(t)=\frac{1}{D}\sum_{i=1}^{D}x_{id}(t), \tag{4}$$

$$x_{id}(t+1)=wx_{id}(t)+c_1r_1(\text{pbest}_{id}-x_{id}(t)), \tag{5}$$

$$x_{id}(t+1)=wx_{id}(t)+c_2r_2(\text{gbest}_d-x_{id}(t)), \tag{6}$$

$$x_{id}(t+1)=wx_{id}(t)+c_3r_3(p_{ad}-x_{id}(t)), \tag{7}$$

其中,式(5)有助于算法的全局开发能力;式(6)有助于算法的局部勘探能力,帮助算法跳出局部最优;式(7)有助于提升算法的收敛速度. 在迭代过程中,算法基于概率选择不同的模式来更新粒子位置.

文献[8]指出采用“$X=X+V$”来更新粒子位置有助于提高算法的局部勘探能力,而“$X=wX+(1-w)V$”有助于提高算法的全局开发能力,基于此本文提出一种自适应粒子位置更新机制,如式(8)、(9)所示.

$$p_i=\frac{\exp(\mathrm{fit}(x_i(t)))}{\exp\left(\frac{1}{N}\sum_{i=1}^{N}\mathrm{fit}(x_i(t))\right)}, \tag{8}$$

$$x_{id}(t+1)=\begin{cases} wx_{id}(t)+(1-w)v_{id}(t+1), & p_i>\mathrm{rand} \\ x_{id}(t)+v_{id}(t+1), & \mathrm{else} \end{cases} \tag{9}$$

式中,fit(·)为粒子的适应度值,N 为种群中粒子个数. 式(8)中 p_i 表示当前粒子适应度值与种群中所有粒子平均适应度值的比值,当 p_i 较大时,当前粒子的适应度值远大于种群中所有粒子平均适应度值,此时应采用“$X=wX+(1-w)V$”更新粒子位置以增强算法的全局开发能力,否则采用“$X=X+V$”来更新粒子位置,来保证算法的局部勘探能力.

2　改进算法

2.1　粒子速度和位置自适应更新策略

本文结合以上分析提出一种新的自适应粒子速度和位置更新策略,如式(10)、(11)所示.

$$\begin{cases} v_{id}(t+1)=wv_{id}(t)+c_1r_1\left(\frac{\mathrm{pbest}_{id}+\mathrm{gbest}_d}{2}-x_{id}(t)\right)+c_2r_2\left(\frac{\mathrm{pbest}_{id}-\mathrm{gbest}_d}{2}-x_{id}(t)\right), \\ x_{id}(t+1)=wx_{id}(t)+(1-w)v_{id}(t+1), \end{cases} \tag{10}$$

$$\begin{cases} v_{id}(t+1)=wv_{id}(t)+c_1r_1(p_{ad}-x_{id}(t))+c_2r_2(\mathrm{gbest}_d-x_{id}(t)), \\ x_{id}(t+1)=x_{id}(t)+v_{id}(t+1), \end{cases} \tag{11}$$

上述自适应策略借鉴文献[8]中 p_i 作为自适应判定条件. 当 $p_i>\delta$ 时,当前粒子的适应度值远大于种群中所有粒子平均适应度值,表明算法处于搜索初期或者当前粒子分布较为分散,此时应采用式(10)更新粒子速度和位置. 式(10)在速度更新项中引入个体最优和群体最优的线性组合,能够使粒子的搜索空间更广,从而提高算法搜索到全局最优解的可能性,而位置更新则采用“$X=wX+(1-w)V$”来提高算法的全局搜索能力;当 $p_i<\delta$ 时,当前粒子的适应度值与种群中所有粒子平均适应度值相差不大,表明算法处于搜索中后期或者当前粒子分布较为集中,此时应采用式(11)更新粒子速度和位置. 在式(11)中,位置更新采用“$X=X+V$”来保证算法局部勘探能力,防止算法在求解复杂多峰函数时陷入局部最优,而将粒子平均维度信息 p_{ad} 引入到粒子速度更新公式中,提高算法的收敛速度[13].

2.2　惯性权重的选取

惯性权重 w 是 PSO 算法的重要参数之一,它可以平衡算法全局开发能力和局部勘探能力,标准 PSO 算法采用线性递减 w,这种线性调整的方式可以在一定程度上平衡算法开发能力与勘探能力,而在面对复杂非线性多维函数优化问题时,算法容易陷入局部最优解. 因此,为更好改善算法性能,一些学者提出惯性权重非线性调整策略. 本文采用文献[8]提出的基于 Logistic 混沌映射非线性变化惯性权重 w.

混沌映射作为一种非线性映射,因其所产生的混沌序列具有良好随机性和空间遍历性,在进化计算中得到广泛应用,其中 Logistic 映射应用较为广泛,它可以产生[0,1]之间的随机数. 式(12)给出了 Logistic 映射的定义,w 的定义如式(13).

$$r(t+1)=4r(t)(1-r(t)),r(0)=\mathrm{rand},\ \text{其中},r_0\neq\{0,0.25,0.75,1\}, \tag{12}$$

$$w(t)=r(t)w_{\min}+\frac{(w_{\max}-w_{\min})t}{T_{\max}}, \tag{13}$$

式中,$w_{\max}=0.9$,$w_{\min}=0.4$;$r(t)$ 是由式(12)迭代产生的随机数. 惯性权重随迭代次数的变化如图 2 所示.

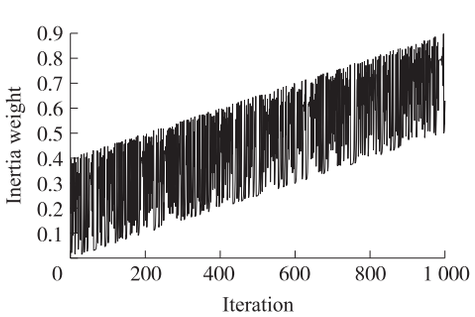

图 2　惯性权重随迭代次数变化图

Fig. 2　The variation of inertia weight with iteration times

2.3 算法实现

算法实现步骤如下:

步骤 1　参数初始化,包括种群规模 N,最大迭代次数 $T_{\max}=1\ 000$,学习因子 c_1 和 c_2,惯性权重 w;并在搜索空间内随机初始化粒子位置、速度;

步骤 2　根据给定目标函数计算所有粒子适应度值;

步骤 3　比较种群中所有粒子的适应度值与其经历过的最优位置适应度值的优劣,若前者更优,则用粒子的当前位置替代粒子的个体历史最优位置;

步骤 4　比较种群中所有个体最优位置的适应度值与整个群体经历过的最优位置的适应度值的优劣,若前者更优,则更新全局最优位置;

步骤 5　根据式(8)计算 p_i,若 $p_i>\delta$,则利用式(10)更新粒子的速度和位置;否则,利用式(11)更新粒子的速度和位置;

步骤 6　判断是否满足终止条件,若满足,则算法终止并输出最优值;否则,转至步骤 2.

3 仿真实验及结果分析

3.1 基本测试函数

为验证所提算法有效性,本文选取 12 个具有不同特性的测试函数[13,15],将本文算法与线性递减惯性权重粒子群优化(LDWPSO)算法[7]、均值粒子群优化(MeanPSO)算法[11]、社会学习粒子群优化(SL-PSO)算法[16]、基于概率分层的简化粒子群优化(PHSPSO)算法[13]、基于终端交叉和转向扰动的粒子群优化(TCSPSO)算法[15]、一种基于自适应策略的改进粒子群优化(MPSO)算法[8]进行对比测试. 12 个测试函数的相关描述如表 1 所示. 其中函数 f_1、f_2、f_3、f_4、f_5 是单峰函数,主要用于测试算法的收敛速度和寻优精度;函数 f_6、f_7、f_8、f_9、f_{10}、f_{11}、f_{12} 是多峰函数,主要用于测试算法跳出局部最优避免“早熟”收敛的能力.

表 1　12 个基本测试函数

Table 1　12 basic test functions

No	Function	Formulation	Range	f_{opt}
f_1	Sphere	$f=\sum_{i=1}^{D}(x_i)^2$	$[-100,100]$	0
f_2	Sum Squares	$f=\sum_{i=1}^{D}(ix_i)^2$	$[-5.21,5.21]$	0
f_3	Sum of Different Power	$f=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	$[-1,1]$	0
f_4	Schwefel 2.22	$f=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]$	0
f_5	Schwefel 1.2	$f=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	$[-100,100]$	0
f_6	Ackley	$f=-20\exp\left(0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i)^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+\mathrm{e}$	$[-32,32]$	0
f_7	Rastrigin	$f=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.21,5.21]$	0
f_8	Noncontinuous Rastrigin	$f=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi y_i)+10]$ $y_i=\begin{cases}x_i, & \lvert x_i\rvert<0.5\\ \dfrac{\mathrm{round}(2x_i)}{2}, & \text{else}\end{cases}$	$[-5.21,5.21]$	0
f_9	Griewank	$f=\frac{1}{4\ 000}\sum_{i=1}^{D}x_i^2-\prod\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$	0
f_{10}	Levy	$f=\sin^2(\pi w_1)+\sum_{i=1}^{D-1}(w_i-1)^2[1+10\sin^2(\pi w_i+1)]+(w_d-1)^2[1+\sin^2(2\pi w_d)]$, $w_i=1+\frac{x_i-1}{4}$	$[-10,10]$	0

续表 1　Table 1 continued

No	Function	Formulation	Range	f_{opt}
f_{11}	Penalized	$f=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{D}U(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1),U_{x_i,a,k,m}=\begin{cases}k(x_i-a)^m,x_i>a\\0,-a\leqslant x_i\leqslant a\\k(-x_i-a)^m,x_i<-a\end{cases}$	$[-50,50]$	0
f_{12}	Penalized2	$f=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{D-1}U(x_i,5,100,4)$ $y_i=1+\frac{1}{4}(x_i+1),U_{x_i,a,k,m}=\begin{cases}k(x_i-a)^m,x_i>a\\0,-a\leqslant x_i\leqslant a\\k(-x_i-a)^m,x_i<-a\end{cases}$	$[-50,50]$	0

3.2　阈值参数 δ 取值对算法性能的影响

文献[8]中阈值 δ 取值为[0,1]之间的随机数,本文认为阈值 δ 随机变化不利于对当前粒子状态准确判断,从而影响算法性能. 本文将进行一组实验来验证此观点,进而确定阈值 δ 最佳取值. 实验选取表 1 中 4 个测试函数($D=100$),算法独立运行 50 次,取最优结果的平均值作为比较,具体结果见表 2. 由表 2 知,δ 取 0.8 时,对于单峰函数 f_4,算法收敛精度最高;而对于多峰函数 f_{10}、f_{12},算法收敛速度最快;对于函数 f_{11},算法的收敛精度要远优于 δ 取 1 和 rand 时的结果. 故本文 δ 取 0.8. 表 2 中括号内数字代表算法求得最优解时的平均迭代次数(取整).

3.3　与其他改进 PSO 算法对比

本文使用 Matlab 软件进行仿真,不同 PSO 算法设置相同种群规模 $N=100$、最大迭代次数 $T_{max}=1\ 000$ 和变量维数 $D=30$,其他参数设置与原文保持一致,具体见表 3. 每个算法独立运行 50 次,记录 50 次运行结果的平均值(Mean value)和标准差(Standard deviation)来评价算法性能,结果见表 4. 为更直观对比各算法求解精度和收敛速度,图 3 给出各算法求解 12 个测试函数时适应度值变化曲线.

表 2　参数 δ 不同取值的优化结果比较

Table 2　Comparison of optimization results with different values of parameter δ

δ	0.6	0.8	1	rand
f_4	2.46e-246	1.15e-252	1.01e-244	1.91e-245
f_{10}	1.50e-32 (110)	1.50e-32 (94)	1.50e-32 (105)	1.50e-32 (115)
f_{11}	4.71e-33	4.71e-33	6.11e-23	8.85e-16
f_{12}	1.35e-32 (112)	1.35e-32 (110)	1.35e-32 (200)	1.35e-32 (130)

表 3　各 PSO 算法的参数设置

Table 3　Parameter setting of each PSO algorithm

Algorithm	Parameter setting
LDWPSO	$w_{max}=0.9,w_{min}=0.4,c_1=c_2=2$
MeanPSO	$w=0.7,c_1=c_2=2$
SL-PSO	A dimension dependent parameter control method is adopted
PHSPSO	$w=0.9,c_1=c_2=2$
TCSPSO	$w_{max}=0.9,w_{min}=0.4,c_1=c_2=2$
MPSO	$w_{max}=0.9,w_{min}=0.4,c_1=c_2=2$
IPSO-PV	$w_{max}=0.9,w_{min}=0.4,c_1=c_2=2,\delta=0.8$

表 4　7 种算法求解 12 个测试函数的结果对比($D=30$)

Table 4　Results comparison of 7 algorithms for 12 test functions($D=30$)

Function	Index	IPSO-VP	LDWPSO	MeanPSO	SL-PSO	PHSPSO	TCSPSO	MPSO
f_1	Mean	0.00e+00	6.63e-02	4.17e-103	4.02e-44	5.84e-184	2.42e-103	1.05e-61
	Std	0.00e+00	4.85e-02	2.41e-102	3.52e-44	0.00e+00	9.56e-103	3.96e-61
f_2	Mean	0.00e+00	1.39e-02	1.08e-103	1.76e-45	2.01e-182	2.76e-105	4.30e-64
	Std	0.00e+00	2.09e-02	4.83e-103	3.43e-45	0.00e+00	1.21e-104	1.97e-63
f_3	Mean	0.00e+00	9.47e-15	3.75e-111	2.38e-104	9.06e-34	4.73e-151	1.41e-186
	Std	0.00e+00	2.69e-14	1.68e-110	1.68e-103	4.69e-33	2.66e-150	0.00e+00
f_4	Mean	1.83e-248	5.81e-01	5.50e-53	2.31e-23	2.07e-92	1.92e-58	1.23e-34
	Std	0.00e+00	4.46e-01	2.87e-52	1.45e-23	5.02e-92	7.09e-58	7.95e-34
f_5	Mean	0.00e+00	1.84e+01	4.38e-102	6.53e-02	2.04e-181	1.70e-78	2.90e+02
	Std	0.00e+00	9.90e+00	2.59e-101	5.92e-02	0.00e+00	6.10e-78	2.04e+02
f_6	Mean	8.88e-16	2.41e+00	8.88e-16	5.58e-15	8.88e-16	8.88e-16	3.59e-15
	Std	0.00e+00	4.86e-01	0.00e+00	1.38e-15	0.00e+00	0.00e+00	1.53e-15
f_7	Mean	0.00e+00	2.69e+01	0.00e+00	1.40e+01	0.00e+00	0.00e+00	1.13e+01
	Std	0.00e+00	8.43e+00	0.00e+00	3.86e+00	0.00e+00	0.00e+00	1.73e+01

续表 4　Table 4 continued

Function	Index	IPSO-VP	LDWPSO	MeanPSO	SL-PSO	PHSPSO	TCSPSO	MPSO
f_8	Mean	0.00e+00	3.81e+01	0.00e+00	1.67e+01	0.00e+00	0.00e+00	1.39e+01
	Std	0.00e+00	1.00e+01	0.00e+00	9.97e+00	0.00e+00	0.00e+00	1.78e+01
f_9	Mean	0.00e+00	1.54e−01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	Std	0.00e+00	7.35e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_{10}	Mean	1.50e−32	8.89e−01	4.85e−03	3.33e−02	2.14e−07	9.92e−07	1.47e−01
	Std	1.38e−47	6.26e−01	3.42e−02	1.16e−01	2.09e−07	1.35e−06	1.12e−01
f_{11}	Mean	1.57e−32	6.82e−01	6.41e−03	1.57e−32	6.48e−09	3.00e−09	8.94e−04
	Std	5.53e−48	6.26e−01	1.23e−02	5.53e−48	9.97e−09	3.32e−09	2.62e−03
f_{12}	Mean	1.35e−32	1.92e−01	2.40e−01	1.35e−32	7.66e−08	6.90e−04	1.14e−02
	Std	1.11e−47	1.60e−01	6.88e−01	1.11e−47	9.72e−08	2.63e−03	2.03e−02

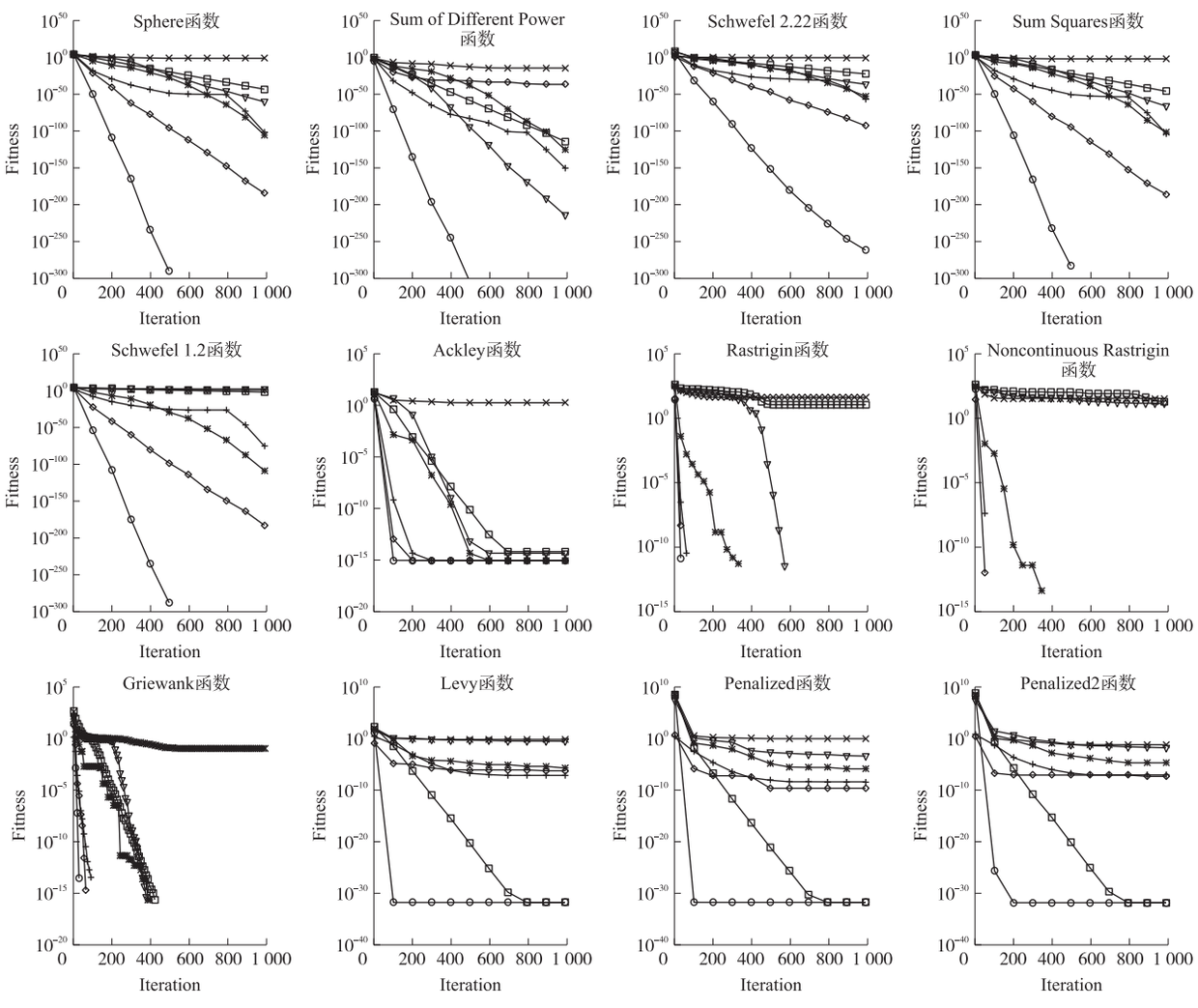

图 3　12 个测试函数的收敛曲线(30 维)

Fig. 3　Convergence curves of 12 test functions(30 dimensions)

分析表 4 和图 3 可知改进算法(IPSO-VP)在 30 维f_1、f_2、f_3、f_4、f_5 5 个单峰函数上,无论是求解精度、收敛速度还是稳定性均优于 6 个对比算法,且在函数f_1、f_2、f_3、f_5 上 IPSO-VP 算法均能求得最优值. 对于多峰函数f_6、f_7、f_8、f_9,虽然 IPSO-VP 与 MeanPSO、PHSPSO、TCSPSO 算法均能求得函数最优值且求解性能稳定,但从图 3 可看出 IPSO-VP 算法收敛速度更快. 对于多峰函数f_{10}、f_{11}、f_{12},IPSO-VP 算法求解性能明显优于除 SL-PSO 算法外的 5 个算法,虽然 SL-PSO 算法的求解精度与 IPSO-VP 算法相同,但从图 3 收敛曲线可看出 IPSO-VP 算法收敛更快,约迭代 100~200 次之后就完成收敛.

为进一步验证 IPSO-VP 算法优越性,将 7 种算法分别在 60 维和 100 维的 12 个测试函数上进行寻优

测试,表 5、表 6 分别记录了各算法在 60 维、100 维的 12 个测试函数上独立运行 50 次所得结果的平均值和标准差.

表 5　7 种算法求解 12 个测试函数结果的平均值和标准差($D=60$)

Table 5　Mean value and standard deviation of 7 algorithms for 12 test functions($D=60$)

Function	Index	IPSO-VP	LDWPSO	MeanPSO	SL-PSO	PHSPSO	TCSPSO	MPSO
f_1	Mean	0.00e+00	1.46e+01	2.26e−104	4.78e−25	6.33e−183	1.04e−99	3.26e+01
	Std	0.00e+00	4.42e+00	1.48e−103	4.01e−25	0.00e+00	4.83e−99	1.62e+02
f_2	Mean	0.00e+00	2.70e+00	5.97e−104	4.48e−26	5.91e−183	7.56e−102	1.24e−52
	Std	0.00e+00	1.19e+00	3.04e−103	5.19e−26	0.00e+00	2.69e−101	8.21e−52
f_3	Mean	0.00e+00	5.55e−14	1.60e−111	5.66e−70	2.58e−34	2.09e−150	1.31e−188
	Std	0.00e+00	1.18e−13	5.51e−111	1.99e−69	1.75e−33	1.15e−149	0.00e+00
f_4	Mean	1.19e−240	6.02e+00	2.25e−52	1.92e−13	1.73e−92	6.13e−58	3.37e−29
	Std	0.00e+00	1.61e+00	7.61e−52	8.79e−14	4.46e−92	1.61e−57	2.28e−28
f_5	Mean	0.00e+00	1.28e+03	4.25e−101	8.67e+03	1.86e−179	1.81e−71	4.32e+03
	Std	0.00e+00	3.52e+02	2.76e−100	3.17e+03	0.00e+00	7.94e−71	2.62e+03
f_6	Mean	8.88e−16	4.63e+00	8.88e−16	1.29e−13	8.88e−16	8.88e−16	4.44e−15
	Std	0.00e+00	6.16e−01	0.00e+00	5.48e−14	0.00e+00	0.00e+00	0.00e+00
f_7	Mean	0.00e+00	6.06e+01	0.00e+00	9.43e+01	0.00e+00	0.00e+00	5.25e+00
	Std	0.00e+00	1.44e+01	0.00e+00	6.15e+01	0.00e+00	0.00e+00	1.83e+01
f_8	Mean	0.00e+00	7.12e+01	0.00e+00	3.36e+02	0.00e+00	0.00e+00	9.35e+01
	Std	0.00e+00	2.97e+01	0.00e+00	3.79e+01	0.00e+00	0.00e+00	8.69e+01
f_9	Mean	0.00e+00	1.13e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	Std	0.00e+00	4.89e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_{10}	Mean	1.50e−32	3.08e+00	1.16e−01	3.33e−02	3.32e−07	3.73e−01	1.68e+00
	Std	1.38e−47	1.53e+00	4.25e−01	1.16e−01	4.69e−07	1.94e−01	2.75e−01
f_{11}	Mean	7.85e−33	2.52e+00	9.03e−02	1.57e−32	7.86e−09	2.38e−04	7.89e+01
	Std	2.76e−48	7.74e−01	3.13e−02	5.53e−48	1.08e−08	1.55e−04	4.88e+02
f_{12}	Mean	1.35e−32	6.16e+01	5.06e+00	1.10e−03	1.26e−07	1.08e+00	2.03e+01
	Std	1.11e−47	2.65e+01	1.04e+00	3.33e−03	1.44e−07	4.28e−01	1.08e+02

表 6　7 种算法求解 12 个测试函数结果的平均值和标准差($D=100$)

Table 6　Mean value and standard deviation of 7 algorithms for 12 test functions($D=100$)

Function	Index	IPSO-VP	LDWPSO	MeanPSO	SL-PSO	PHSPSO	TCSPSO	MPSO
f_1	Mean	0.00e+00	7.51e+02	3.22e−101	4.11e−15	2.11e−182	1.30e−98	1.46e+03
	Std	0.00e+00	1.63e+02	1.53e−100	2.70e−15	0.00e+00	4.60e−98	1.43e+03
f_2	Mean	0.00e+00	1.19e+02	2.60e−103	4.78e−16	3.91e−182	3.83e−99	1.71e−46
	Std	0.00e+00	2.20e+01	1.35e−102	4.13e−16	0.00e+00	1.56e−98	1.21e−45
f_3	Mean	0.00e+00	1.03e−10	1.02e−110	2.00e−39	1.79e−33	4.31e−151	3.02e−187
	Std	0.00e+00	3.00e−10	4.90e−110	8.75e−39	1.01e−32	2.89e−150	0.00e+00
f_4	Mean	3.11e−243	3.40e+01	6.76e−52	3.73e−08	1.68e−92	2.74e−57	1.36e−25
	Std	0.00e+00	4.67e+00	2.28e−51	2.75e−08	5.15e−92	7.56e−57	9.36e−25
f_5	Mean	0.00e+00	1.21e+04	1.66e−101	1.09e+05	2.18e−174	6.88e−68	1.87e+04
	Std	0.00e+00	3.34e+03	6.56e−101	1.51e+04	0.00e+00	2.58e−67	9.28e+03
f_6	Mean	8.88e−16	8.11e+00	8.88e−16	1.05e−08	8.88e−16	8.88e−16	4.44e−15
	Std	0.00e+00	7.23e−01	0.00e+00	3.05e−09	0.00e+00	0.00e+00	0.00e+00
f_7	Mean	0.00e+00	2.35e+02	0.00e+00	6.42e+02	0.00e+00	0.00e+00	2.87e+02
	Std	0.00e+00	2.93e+01	0.00e+00	2.31e+02	0.00e+00	0.00e+00	1.33e+02
f_8	Mean	0.00e+00	3.61e+02	0.00e+00	8.33e+02	0.00e+00	0.00e+00	3.41e+02
	Std	0.00e+00	6.87e+01	0.00e+00	3.67e+01	0.00e+00	0.00e+00	1.18e+02
f_9	Mean	0.00e+00	8.01e+00	0.00e+00	9.86e−04	0.00e+00	0.00e+00	0.00e+00
	Std	0.00e+00	1.86e+00	0.00e+00	3.15e−03	0.00e+00	0.00e+00	0.00e+00
f_{10}	Mean	1.50e−32	8.34e+00	5.54e+00	1.72e+00	8.84e−07	3.37e+00	4.60e+00
	Std	1.38e−47	2.73e+00	1.29e+00	1.42e+00	1.19e−06	4.24e−01	3.32e−01
f_{11}	Mean	4.71e−33	7.93e+00	1.41e−01	8.71e−03	5.33e−09	1.04e−02	3.34e+04
	Std	6.91e−49	3.03e+00	2.90e−02	2.82e−02	9.06e−09	4.20e−03	1.58e+05
f_{12}	Mean	1.35e−32	1.64e+02	8.44e+00	1.54e−03	2.46e−07	6.84e+00	2.81e+04
	Std	1.11e−47	3.44e+01	1.55e+00	3.85e−03	2.69e−07	5.74e−01	7.78e+04

从表5和表6可看出,对于函数f_1、f_2、f_3、f_5、f_7、f_8、f_9,无论是60维还是100维,IPSO-VP算法均能求得函数最优解;f_{10}、f_{11}、f_{12} 3个多峰函数的局部极小值个数会随着变量维数的增加呈指数级增长[17],求解难度增大,但IPSO-VP算法对这3个函数的求解结果依然有很高的精度,且明显优于其他6个算法. IPSO-VP、MeanPSO、PHSPSO、TCSPSO算法在函数f_6、f_7、f_8、f_9上的求解结果相同,为进一步比较这4种算法优劣,图4给出了4种算法求解100维的4个多峰函数的适应度值变化曲线,可以看出IPSO-VP算法收敛速度更快.

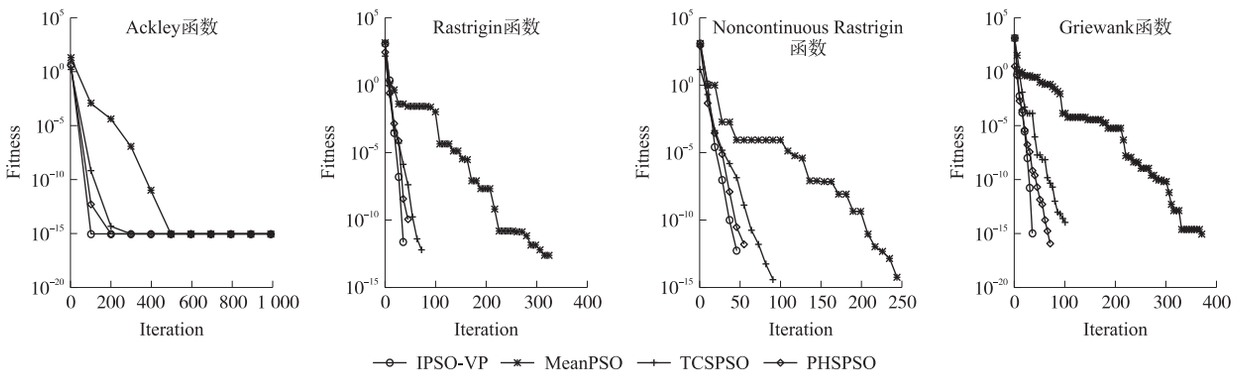

图4　4个测试函数的收敛曲线(100维)

Fig. 4　Convergence curves of four test functions(100 dimensions)

3.4　算法时间复杂度分析

标准PSO算法时间复杂度为$O(nmD)$[18],其中n为粒子数,m为算法迭代次数,D为问题维数,本文IPSO-VP算法利用式(10)、(11)来更新粒子速度和位置,与标准PSO算法相比仅多一项自适应判定,因此,改进策略并未增加算法时间复杂度,本文算法时间复杂度仍为$O(nmD)$. 并且,由图3收敛曲线可看出,在求解精度相同情况下,本文算法所需迭代次数更少. 综上分析,当种群规模、问题维数和迭代次数相同时,本文算法的时间复杂度并未增加.

4　结论

本文提出一种改进粒子速度和位置更新公式的粒子群算法,算法中引入一种自适应粒子速度和位置更新策略,同时采用基于Logistic混沌的非线性惯性权重,实验结果表明,改进算法具有较快的收敛速度和较高的寻优精度,同时改进算法解决维数较高的多峰函数问题具有良好性能和潜在应用价值. 未来工作中,应用改进算法求解实际优化问题是值得研究的内容.

[参考文献]

[1] KENNEDY J,EBERHART R. Particle swarm optimization[C]//Proceedings of ICNN'95-International Conference on Neural Networks,Perth,WA,Australia. IEEE,1995,4:1942-1948.

[2] LIAN J,YU W,XIAO K,et al. Cubic spline interpolation-based robot path planning using a chaotic adaptive particle swarm optimization algorithm[J]. Mathematical problems in engineering,2020(3):1-20.

[3] 居佳琪,王琦,唐小波,等. 基于双重粒子群算法的电动汽车参与配网优化调度[J]. 南京师范大学学报(工程技术版),2018,18(1):11-23.

[4] 程泽,董梦男,杨添剀,等. 基于自适应混沌粒子群算法的光伏电池模型参数辨识[J]. 电工技术学报,2014,29(9):245-252.

[5] 韦苗苗,江铭炎. 基于粒子群优化算法的多阈值图像分割[J]. 山东大学学报(工学版),2005,35(6):118-121.

[6] 王凌,刘波. 微粒群优化与调度算法[M]. 北京:清华大学出版社,2008.

[7] SHI Y,EBERHART R. A modified particle swarm optimizer[C]//1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence(Cat. No. 98TH8360),Anchorage,AK,USA. 1998,4:69-73.

[8] LIU H,ZHANG X W,TU L P. A modified particle swarm optimization using adaptive strategy[J]. Expert systems with appli-

cations,2020,152:113353.
[9] 姜建国,田旻,王向前,等. 采用扰动加速因子的自适应粒子群优化算法[J]. 西安电子科技大学学报,2012,39(4):74-80.
[10] 顾明亮,李旻. 基于动态调整惯性权重的混合粒子群算法[J]. 计算机与现代化,2018(6):25-29.
[11] DEEP K,BANSAL J C. Mean particle swarm optimisation for function optimization[J]. International journal of computational intelligence studies,2009,1(1):72-92.
[12] 胡旺,李志蜀. 一种更简化而高效的粒子群优化算法[J]. 软件学报,2007,18(4):861-868.
[13] LIU H R,CUI J C,LU Z D,et al. A hierarchical simple particle swarm optimization with mean dimensional information[J]. Applied soft computing,2019,76:712-725.
[14] 陆松建,司伟立,韩娟,等. 逃逸均值简化粒子群优化算法[J]. 计算机工程与设计,2020,41(9):2623-2629.
[15] ZHANG X W,LIU H,ZHANG T,et al. Terminal crossover and steering-based particle swarm optimization algorithm with disturbance[J]. Applied soft computing,2019,85:105841.
[16] CHENG R,JIN Y. A social learning particle swarm optimization algorithm for scalable optimization[J]. Information sciences,2015,291:43-60.
[17] GAO W F,LIU S Y,HUANG L L. Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique[J]. Communications in nonlinear science & numerical simulation,2012,17(11):4316-4327.
[18] 吴润秀,孙辉,朱德刚,等. 具有高斯扰动的最优粒子引导粒子群优化算法[J]. 小型微型计算机系统,2016,37(1):146-151.

[责任编辑:丁　蓉]